UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

DESARROLLO DE UN SISTEMA DE BÚSQUEDAS EN VOZ BASADAS EN EJEMPLOS

María Begoña Aguirre Villar
Tutor: Doroteo Torre Toledano

Junio 2017

DESARROLLO DE UN SISTEMA DE BÚSQUEDAS EN VOZ BASADAS EN EJEMPLOS

María Begoña Aguirre Villar
TUTOR: Doroteo Torre Toledano

AUDIAS – AUdio, Data Intelligence And Speech
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2017

Resumen (castellano)

En la actualidad existen diversas técnicas para implementar un sistema de búsqueda en voz. En este TFG se describe el desarrollo de un sistema de búsqueda en voz basadas en ejemplos usando el método QbE - STD (*Query-by-Example – Spoken Detection Term)*. El sistema se ajusta a las condiciones de la evaluación ALBAYZIN 2016, es decir, se usan los mismos recursos (archivos) que son proporcionados para los participantes y se despliega uno de los sistemas propuestos.

El sistema QbE – STD tiene como objetivo encontrar un término (consulta) en un repositorio (documento). Sin embargo, en lugar de usar palabras o fonemas en formato de texto como términos de búsqueda, el sistema emplea fragmentos de audio como términos de búsqueda. Estos fragmentos, son las consultas y han de ser pre-procesadas. La fase de pre-procesamiento se basa esencialmente en la extracción de características. Conjuntamente los documentos han de pasar por este mismo proceso.

La técnica adoptada para la extracción de características debido a sus buenos resultados es la de probabilidades a posteriori de fonemas. Estas son obtenidas a través de un reconocedor fonético.

Por otro lado, otra fase fundamental en el desarrollo del sistema es la implementación de un algoritmo de comparación. Se ha implementado el algoritmo S-DTW (DTW de Subsecuencias) el cual es una variante del algoritmo original de alineamiento temporal dinámico (DTW). El algoritmo decide si la consulta se encuentra en el documento o no. En caso afirmativo, nos proporciona los instantes de tiempo en los cuales comienza y termina una posible detección de la consulta dentro de la base de datos. En añadidura se devuelve la puntuación de la detección.

Finalmente; se realizan una serie de pruebas y resultados para evaluar la eficacia del sistema.

.

.

Palabras clave (castellano)

Búsqueda mediante ejemplo, Reconocimiento de voz, Alineamiento Temporal Dinámico, Extracción de características, Probabilidad a posteriori de un fonema,

Abstract (English)

Nowadays, there are available several techniques for the development of a voice search system. In this Bachelor Thesis, we describe a voice search system based on examples employing the QbE – STD (Query-by-Example – Spoken Detection Term) approach. This system satisfies the ALBAYZIN 2016 evaluation conditions, in other words, the system makes use of the same resources (audio files) that were provided for the participants. Furthermore, one of the proposed systems is deployed.

The aim of the QbE - STD system is to detect a term (query) in an audio database. Nevertheless, instead of using words or some phonemes in a text format as search terms, the system uses audio fragments as search terms. This audio fragments are the queries, and they must be pre-processed. The pre-processing step is essentially based on the feature extraction. Jointly, the audio database must go through this same process.

The phoneme posteriorgrams is the feature extraction approach, this choice is due to its excellent results on the evaluation. These are obtained using phoneme decoders.

On the other hand, it is indispensable the implementation of a searching algorithm. The chosen algorithm is the subsequence DTW, a variant of the classic DTW approach. The algorithm decides whether the query is in the audio database or not. If the query is found, it returns the boundaries (the beginning and the end) of the query in the audio database. In addition, the detection score is returned.

Finally, several tests and results are performed to evaluate the system's performance.

Keywords (inglés)

Query-by-Example, Speech Recognition, Dynamic Time Warping, Feature Extraction, Phoneme posteriorgrams

Agradecimientos

En primer lugar, agradecer a mi familia por su apoyo incondicional y su confianza depositada en todo momento. En especial a mis padres, a ellos les debo haber llegado hasta aquí.

A mi compañero en este viaje; por ser mi punto de apoyo, por haberme ayudado a superar todos los obstáculos y ser el mejor equipo eficaz.

A mi tutor Doroteo; por haberme dado la oportunidad de realizar este trabajo, por orientarme cuando lo he necesitado y por su empeño para que todo saliera de la mejor forma posible.

Este trabajo es la culminación de años de dedicación y esfuerzo, por ello, querría agradecer a todos aquellos que de una manera u otra me han ayudado a llegar hasta aquí.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

En noviembre de 2016 tuvo lugar la evaluación competitiva de sistemas de búsqueda en voz (ALBAYZIN 2016 *Search-on-Speech*) apoyada por la Red Temática Española sobre Tecnología del Habla (RTTH) y fue organizada por FOCUS S.L. Y ATVS - Grupo de Reconocimiento Biométrico de la Universidad Autónoma de Madrid.

Esta evaluación tenía como objetivo encontrar una lista de términos / consultas en ficheros de audio de una duración considerable. En otras palabras, esta evaluación se centra en la recuperación de los archivos de audio que contienen cualquiera de esos términos o consultas. El desarrollo de este sistema se podía implementar de dos maneras diferentes: mediante el uso de la detección de Términos Hablados (STD) y/o el uso del sistema de búsqueda mediante ejemplos (QBE). [1]

En este TFG, el sistema a desarrollar es el sistema de búsqueda mediante ejemplos. Para dicha evaluación no se había realizado ningún sistema de búsqueda mediante ejemplo por parte del grupo ATVS. Por tanto, desarrollaremos un primer sistema de búsquedas mediante ejemplo (*Query-by-Example*). El desarrollo de este sistema se realiza con los mismos recursos que fueron entregados para la evaluación competitiva, es decir, se usan los mismos ficheros de audio como consultas y las mismas bases de datos.

1.2 Objetivos

El objetivo principal es desarrollar un sistema básico de búsquedas mediante ejemplo (*Query-by-Example*) que permita buscar segmentos de audio similares a un ejemplo dado.

Asimismo, como se ha mencionado anteriormente, el objetivo que persigue este trabajo es realizar una primera evaluación del sistema. Esta se realizará usando las *queries* (consultas) y los archivos de audio que fueron proporcionados a los participantes de la evaluación competitiva.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

1. **Introducción**
    - Motivación
    - Objetivo
2. **Estado del arte**
    - Reconocimiento del habla
        - Contexto Histórico
        - Extracción de características
        - Métodos / algoritmos
    - Sistema de búsquedas en voz
    - Usos de QbE
3. **Diseño**
    - Material
    - Pre-procesamiento:
        1. Reconocedor BRNO
        2. HTK
    - Algoritmo S-DTW
4. **Desarrollo**
    - Algoritmo S-DTW:
        1. Réplica del mejor sistema
        2. Diseño del algoritmo
        3. Salida del algoritmo
5. **Integración, pruebas y resultados**
    - Resultados después de aplicar algoritmo
    - Comparativa con los resultados de ALBAYZIN
6. **Conclusiones y trabajos futuros**
    - Funcionamiento del algoritmo desarrollado, eficiencia, resultados…
    - Trabajos futuros como continuación de este Trabajo de Fin de Grado.

2 Estado del arte

Actualmente se dispone de un considerable volumen de datos de voz heterogéneos almacenados en repositorios de audio y audiovisuales. Este está además aumentando de forma exponencial, lo que da lugar a la necesidad de métodos eficientes para recuperar dicha información.

2.1 Reconocimiento del habla

Un sistema de reconocimiento de voz es una herramienta computacional capaz de procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes que actúan sobre un proceso. Se puede dividir en dos fases: el entrenamiento y el reconocimiento.

Durante la fase de entrenamiento, se genera un vector de entrenamiento para cada palabra pronunciada por el usuario. Los vectores de entrenamiento extraen las características espectrales para separar diferentes clases de palabras. Cada vector de entrenamiento puede servir como plantilla para una sola palabra o una clase de palabra. Estos vectores de entrenamiento (patrones) se almacenan en una base de datos para uso posterior en la fase de reconocimiento.

En la fase de reconocimiento una vez extraídas las características, se realiza una comparación directa entre el vector característico asociado a la señal de voz desconocida (a reconocer) y todos los posibles patrones aprendidos en la etapa de entrenamiento, de manera a determinar el mejor ajuste de acuerdo a algún criterio.

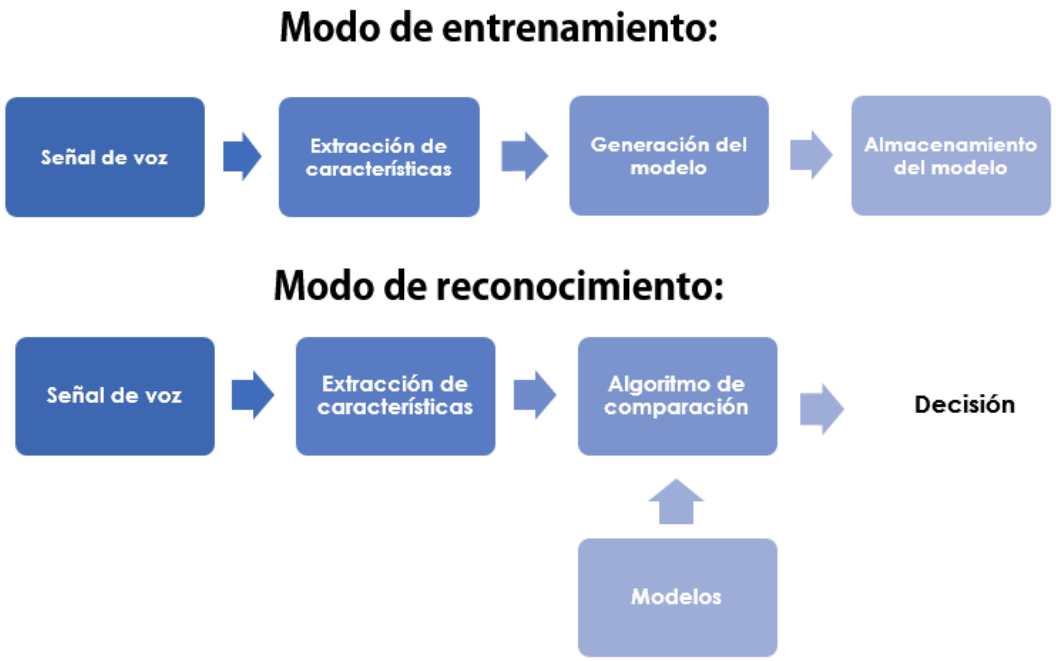

Figura Estado del arte - 1: Fases reconocimiento del habla

2.1.1 Contexto histórico

Los primeros sistemas de reconocimiento de voz sólo podían entender dígitos aislados. Dada la complejidad del lenguaje humano, los inventores e ingenieros se centraron inicialmente en los números.

Bell Laboratories diseñó en 1952 el sistema "Audrey", que reconocía dígitos hablados por una sola voz con un vocabulario de diez palabras [2]. Diez años más tarde, IBM demostró en la Feria Mundial de 1962 su máquina "Shoebox", que podía entender 16 palabras habladas en inglés.

Durante la década de los 60, los investigadores soviéticos inventaron el algoritmo dinámico de la deformación del tiempo (DTW) y lo utilizaron para crear un reconocedor capaz de funcionar en un vocabulario de 200 palabras. El algoritmo DTW [3] procesó la señal de voz dividiéndola en tramas cortas, por ejemplo, 10 ms, y procesando cada trama como una sola unidad. Aunque DTW sería sustituido por algoritmos posteriores, la técnica de dividir la señal en marcos continuaría.

Seguidamente, Leonard Baum desarrolló las matemáticas de las cadenas de Márkov en el Instituto de Análisis de Defensa. En CMU, los estudiantes de Raj Reddy, James Baker y Janet Baker comenzaron a usar el Modelo de Márkov Oculto (HMM) para el reconocimiento del habla. James Baker había aprendido acerca de los Modelos de Márkov Ocultos de un trabajo de verano en el Instituto de Análisis de Defensa durante su educación de pregrado. [4]

El uso de HMM permitió a los investigadores combinar diferentes fuentes de conocimiento, como la acústica, el lenguaje y la sintaxis, en un modelo probabilístico unificado.

La tecnología de reconocimiento de voz avanzó en la década de 1970, gracias al interés y la financiación del Departamento de Defensa de los Estados Unidos. El programa DARPA, de 1971 a 1976, fue uno de los más grandes en la historia del reconocimiento del habla y, entre otras cosas, fue responsable del sistema de comprensión del habla "Harpy" de Carnegie Mellon. Harpy podía entender 1011 palabras, aproximadamente el vocabulario promedio de un niño de tres años de edad.

Bajo la dirección de Fred Jelinek, IBM creó una máquina de escribir activada por voz llamada Tangora, que podría manejar un vocabulario de 20.000 palabras a mediados de los años 80. [5] El enfoque estadístico de Jelinek pone menos énfasis en emular la forma en que el cerebro humano procesa y entiende el habla a favor del uso de técnicas de modelado estadístico como HMM.

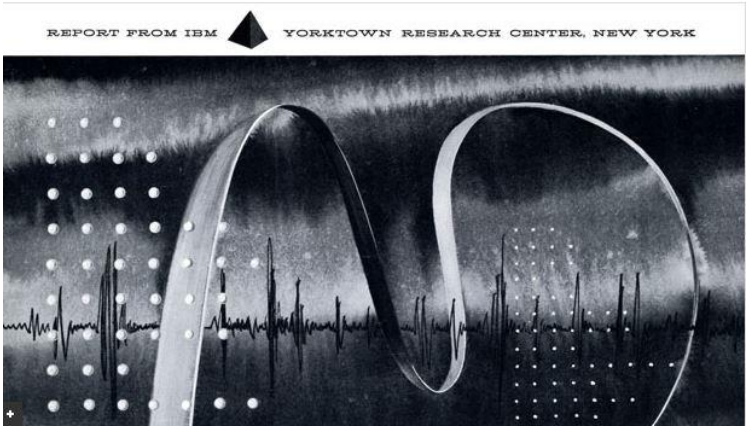

Figura Estado del arte - 2: Anuncio reconocimiento de voz "Tangora"
Fuente: [5]

En los años 90, finalmente llegaron computadoras con procesadores más rápidos y el software de reconocimiento de voz se volvió asequible para la gente común. En este punto, el vocabulario del sistema de reconocimiento de voz comercial típico era más grande que el vocabulario humano promedio. El ex alumno de Raj Reddy, Xuedong Huang, desarrolló el sistema Sphinx-II en CMU. El sistema Sphinx-II fue el primero en hacer un vocabulario extenso, independiente, con reconocimiento de voz continuo y tuvo el mejor desempeño en la evaluación de 1992 de DARPA. Huang continuó fundando el grupo de reconocimiento de voz en Microsoft en 1993. El estudiante de Raj Reddy, Kai-Fu Lee, se unió a Apple donde, en 1992, ayudó a desarrollar un prototipo de interfaz de voz para el ordenador de Apple conocido como Casper.

En 2000, Lernout & Hauspie adquirió Dragon Systems y fue líder de la industria hasta que un escándalo de contabilidad puso fin a la compañía en 2001. En 2001, el reconocimiento de voz por ordenador había alcanzado el 80 por ciento de precisión. Los sistemas de reconocimiento funcionaron bien cuando el universo lingüístico era limitado, pero seguían "adivinando", con la ayuda de modelos estadísticos, entre palabras similares y el conocido universo lingüístico siguió creciendo a medida que crecía Internet.

En resumen, el cuello de botella con el reconocimiento de voz siempre ha sido la disponibilidad de datos, y la capacidad de procesarlo de manera eficiente. La aplicación de Google añade, a su análisis, los datos de miles de millones de consultas de búsqueda, para predecir mejor lo que probablemente se está diciendo. En 2010, Google añadió "reconocimiento personalizado" a la Búsqueda por voz en teléfonos *Android*, para que el software pudiera registrar las búsquedas de voz de los usuarios y producir un modelo de voz más preciso. La compañía también añadió *Voice Search* a su navegador Chrome a mediados de 2011. El sistema de búsqueda de voz en inglés de Google ahora incorpora 230 mil millones de palabras de las consultas reales de los usuarios.

2.1.2 Extracción de características

La voz es una señal continua y variante en el tiempo y, por lo tanto, es difícil de analizar en el dominio temporal. Así que la extracción de características implica el análisis de la señal de voz.

En términos generales, las técnicas de extracción de características se clasifican en análisis temporal y análisis espectral. En el análisis temporal, la forma de onda de la voz se utiliza para el análisis. En el análisis espectral se utiliza la representación espectral de la señal de voz para el análisis.

Haciendo uso de estas técnicas, la señal de entrada se transforma en una secuencia discreta de parámetros, concretamente en una secuencia de vectores de características. El módulo de extracción de características es el encargado de extraer estos vectores. Existen varios métodos para extraer dichos vectores; caben destacar: el uso de coeficientes PLP, el uso de coeficientes MFCC o probabilidades a posteriori de los fonemas dado su eficiencia. Sin embargo, existen otros métodos como los espectrogramas o características de cuello de botella (MLP). Los métodos mencionados anteriormente serán descritos a continuación.

2.1.2.1 Probabilidades a posteriori de los fonemas

Las probabilidades a posteriori se han utilizado principalmente como puntuaciones acústicas locales (medidas) o como características acústicas en sistemas ASR. Métodos como el modelo oculto de Márkov o las redes neuronales [6] fueron de los primeros en hacer uso de las probabilidades a posteriori tratándolas como puntuaciones locales.

Para la extracción de estas características (probabilidades) es necesario el uso de un reconocedor de fonemas. El objetivo de este es la transcripción de una cadena de vectores de características a una serie de unidades fonéticas. El reconocedor recibe como parámetro de entrada una señal de voz (audio) y devuelve un vector que proporciona la probabilidad de ocurrencia de cada fonema para cada trama de audio analizada. [7]

Cada reconocedor es capaz de reconocer N unidades fonéticas. Aunque para cada idioma varíen el número de fonemas, el número de fonemas de la mayoría de todos los idiomas está en un rango entre 20 y 60.

Suponemos un reconocedor de fonemas que incluye *N* unidades fonéticas y que cada una de ellas usa un modelo de *S* estados. Tras haber hecho uso del reconocedor, se asume que este mismo nos devuelve la probabilidad a posteriori de cada de estado $s\ (1 \leq s \leq S)$ de cada fonema $i\ (1 \leq i \leq N)$ en cada trama t, $p\ (i \mid s, t)$.

El tamaño del vector de características se obtiene multiplicando el número de unidades fonéticas (*N*) del reconocedor por el número de estados que usa cada una de las unidades fonéticas. Por consiguiente, para calcular estas características, es decir, la probabilidad a posteriori de cada fonema *i* en cada trama *t* se calcula sumando las probabilidades de cada uno de sus estados:

$$p(i|t) = \sum_{\forall s} p(i|s,t)$$

De esta manera, obtenemos un vector de longitud N, el cual contiene las características requeridas (probabilidades a posteriori). Este vector puede estar en formato binario, es decir, formato ilegible. Dado que la siguiente fase del sistema de reconocimiento es el algoritmo de comparación es necesario que este vector pueda ser legible.

.

Una técnica que fue pionera en aplicar el uso de probabilidades a posteriori como características es Tandem [8]. Para cada instante de habla (es decir, aproximadamente cada 10 ms en un sistema ASR típico), la técnica Tandem deriva un vector de probabilidades a posteriori de estados de cualquier evidencia relevante que presente a su entrada y las añade al vector de características local convencional (ej. MFCC o PLP).

2.1.2.2 Coeficientes PLP

El análisis de PLP se basa en el análisis a corto plazo en el espectro del habla y en el análisis predictivo lineal. PLP modifica el espectro a corto plazo del habla mediante varias transformaciones psicoacústicas. De hecho, PLP es una combinación de Predicción Lineal (LP) y Transformada Discreta de Fourier (DFT).

Para implementar este método son necesarios una concatenación de pasos a realizar con el fin de extraer las caracteriticas. Los pasos incluidos en la siguiente figura muestran cómo convertir una señal de voz en características de PLP como introdujo Hermansky.

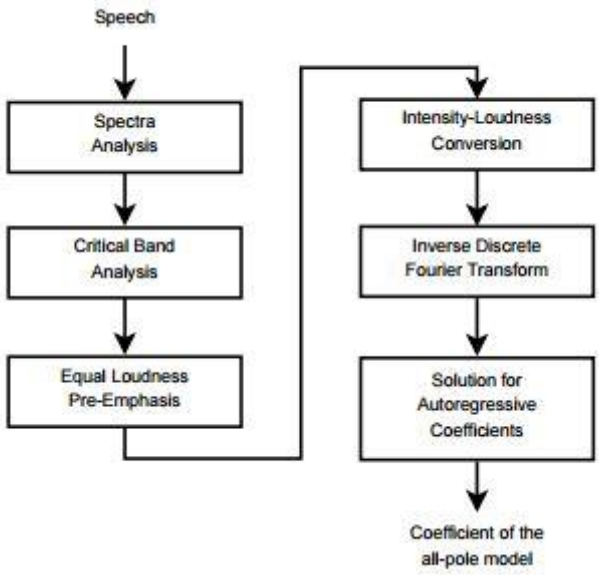

Figura Estado del arte - 3: Pasos implementación del método (PLP)
Fuente: [9]

2.1.2.3 Coeficientes MFCC

Los Mel Frequency Cepstral Coefficients (Coeficientes Cepstrales en las Frecuencias de Mel) o MFCCs son coeficientes para la representación del habla basados en la percepción auditiva humana.

Al igual que el método anterior, para implementar dicho método es necesario seguir unos pasos descritos en la figura siguiente y explicados más adelante.

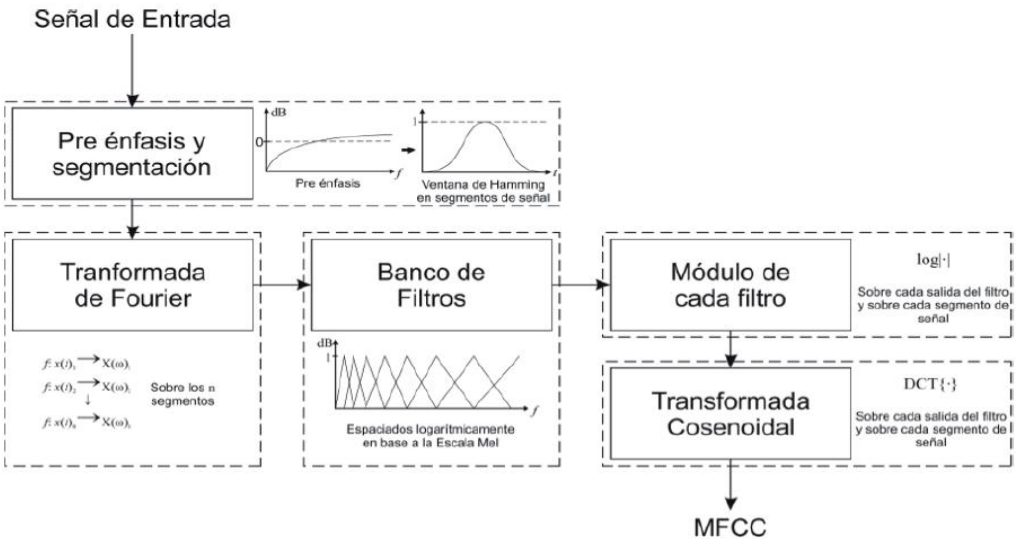

Figura Estado del arte - 4: Pasos implementación del método (MFCC)
Fuente: [10]

A continuación, una breve explicación de los pasos descritos en la figura [11]:

1. Preénfasis y segmentación: En este paso, la señal de entrada se pasa a través de un filtro que enfatiza las frecuencias más altas. Es decir, aumentará la energía de la señal a mayor frecuencia. La señal de voz se segmenta en pequeños bloques de duración de 20-30 ms conocidos como fragmentos.

2. Transformada de Fourier: a cada fragmento se aplica la DFT (convertir del dominio del tiempo en dominio de la frecuencia). De esta manera se consigue hallar la potencia espectral de la señal.

3. Banco de Filtros: después de haber hallado el espectro en el paso 2, se aplica el banco de filtros correspondientes a la Escala Mel, es decir, multiplicar el espectro por un conjunto de N filtros paso banda triangular con el fin de obtener un espectro de magnitud suave.

4. Módulo de cada filtro: se toma logaritmo sobre la salida del banco de filtros.

5. Transformada Discreta del Coseno (DCT): Sobre el paso anterior se aplica la transformada de tipo coseno discreta; obteniendo así, los coeficientes cepstrales en escala-mel.

2.1.3 Métodos de reconocimiento

Tanto el modelado acústico como el modelado de lenguaje conforman una parte importante de los algoritmos de reconocimiento de voz. El modelado del lenguaje también se utiliza en muchas otras aplicaciones de procesamiento del lenguaje natural, como la clasificación de documentos o la traducción automática estadística.

Actualmente existen diferentes métodos o algoritmos de reconocimiento del habla. Estos se han desarrollado a lo largo del tiempo y evolucionando dando lugar a algoritmos con mejoras de eficiencia y eficacia.

2.1.3.1 Alineamiento Temporal Dinámico

El Alineamiento Temporal Dinámico o DTW (*Dynamic Time Warping*) es un método para encontrar una alineación óptima entre dos secuencias dependientes del tiempo que se encuentran bajo ciertas restricciones. Además, esta falta de alineamiento no obedece a una ley fija (p. e., un retardo constante), sino que se da de forma heterogénea, produciéndose así variaciones localizadas que aumentan o disminuyen la duración del tramo de análisis.

Originalmente, este método se ha utilizado para comparar diferentes patrones de habla en el reconocimiento automático de voz. En campos como la extracción de datos y la recuperación de información, DTW se ha aplicado exitosamente para hacer frente de manera automática a las deformaciones de tiempo a velocidades diferentes asociadas con los datos dependientes del tiempo.

La explicación del alineamiento temporal dinámico, así como su variante (S-DTW) han sido obtenidas del libro [12], haciéndose a continuación un resumen de los puntos más relevantes para este TFG.

El objetivo de este algoritmo es comparar dos secuencias, $X = \{x_1, x_2, ..., x_N\}$ de longitud $N \in \mathbb{N}$ y $Y = \{y_1, y_2, ..., y_M\}$ de longitud $M \in \mathbb{N}$. Estas secuencias pueden ser señales discretas o secuencias de características muestreadas en puntos equidistantes en el tiempo. A continuación, en la Figura 5 se puede observar el alineamiento de dos secuencias dependientes del tiempo. Los puntos alineados están indicados por flechas.

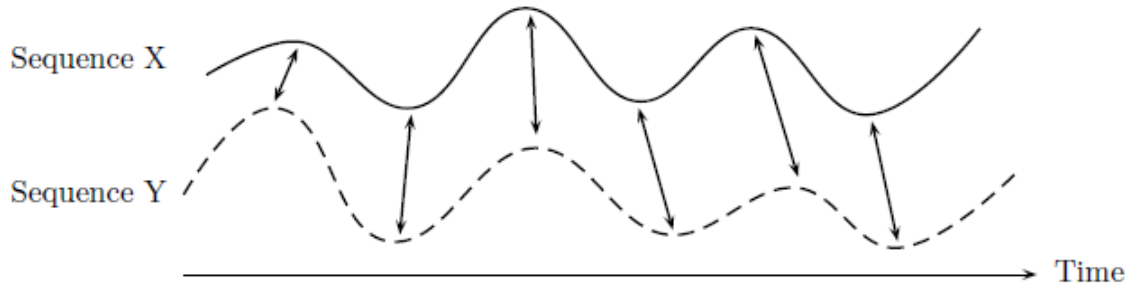

Figura Estado del arte - 5: Alineamiento entre dos secuencias (DTW).
Fuente: Reproducción de [12]

Se fija un espacio característico que denominamos con este carácter $\mathscr{F}$. En consecuencia, tras haber fijado dicho espacio, x_n, $y_m \in \mathscr{F}$ para n ∈ [1: N] y m ∈ [1:M]. Para poder comparar dos características diferentes ($x, y \in \mathscr{F}$) se necesita una medida de coste local, a veces también denominada medida de distancia local, que se define como una función.

$$c: \mathscr{F} \times \mathscr{F} \rightarrow \mathbb{R} \geq 0.$$

Típicamente, $c(x, y)$ es un valor pequeño (bajo coste) si x e y son similares entre sí, y de lo contrario $c(x, y)$ es un valor grande (alto coste). Evaluando la medida de coste local para cada par de elementos de las secuencias X e Y, se obtiene la matriz de coste C ∈ ℝ (NxM) definida por $C(n, m) := c(x_n, y_m)$.

A continuación, se muestra una representación de una matriz de coste de dos secuencias de valor real *X* (eje vertical) e *Y* (eje horizontal) utilizando la distancia Manhattan (valor absoluto de la resta) como medida de coste local c. Las regiones de bajo coste se representan mediante el uso de colores oscuros y las regiones de alto coste se representan mediante el uso de colores claros.

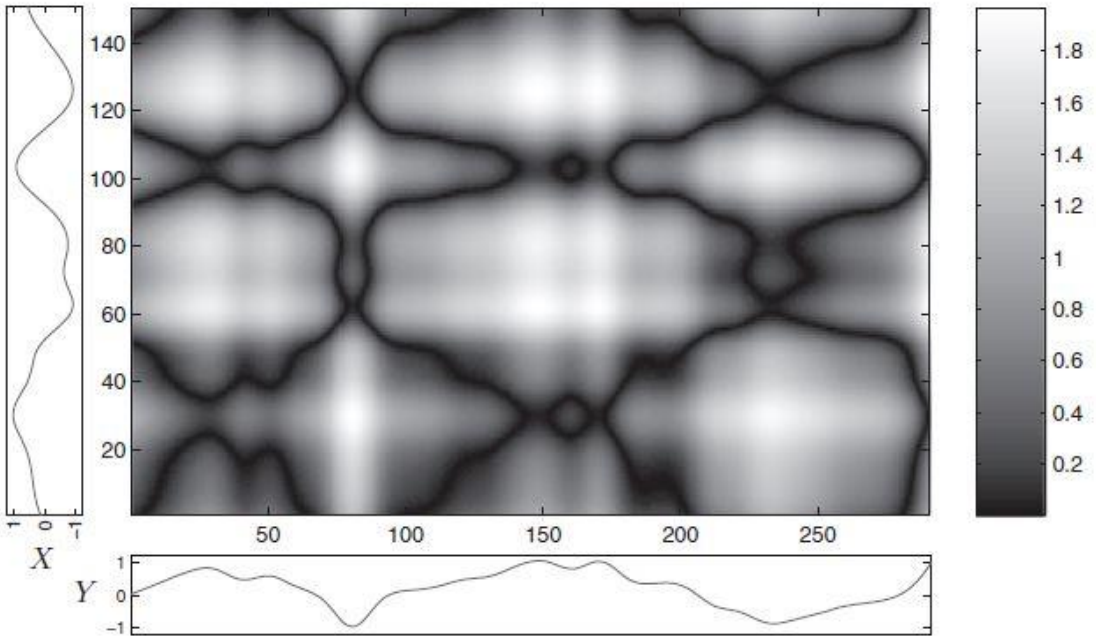

Figura Estado del arte - 6: Representación de la matriz de coste (DTW).
Fuente: Reproducción de [12]

Por consiguiente, el objetivo es hallar una alineación entre X e Y que tenga un coste total mínimo. Intuitivamente, tal alineamiento óptimo se extiende a lo largo de un "valle" de bajo coste dentro de la matriz de coste C, véase la Fig. 5 para una ilustración. La siguiente definición formaliza la noción de alineación.

2.1.3.1.1 Definición

Un camino optimo (N, M) es una secuencia $p = \{p_1, p_2 \ldots p_L\}$ con $p_\ell = (n_\ell, m_\ell) \in [1: N] \times [1: M]$ siendo $\ell \in [1: L]$ que satisface las tres condiciones siguientes:

i. *Condición de frontera:* $p_1 = (1, 1)$ y $p_L = (N, M)$.
ii. *Condición de monotonía:* $n_1 \leq n_2 \leq \ldots \leq n_L$ y $m_1 \leq m_2 \leq \ldots \leq m_L$
iii. *Condición del tamaño de paso:* $p_{\ell+1} - p_\ell \in \{(1, 0), (0, 1), (1, 1)\}$ para $\ell \in [1: L - 1]$.

Obsérvese que la condición de tamaño de paso (iii) implica la condición de monotonía (ii), que sin embargo ha sido citada explícitamente por razones de claridad. Un camino óptimo (N, M) con $p = \{p_1, p_2, \ldots, p_L\}$ define una alineación entre dos secuencias $X = \{x_1, x_2, \ldots, x_N\}$ e $Y = \{y_1, y_2, \ldots, y_M\}$ asignando el elemento x_{n_ℓ} de X y el elemento y_{m_ℓ} de Y. La condición de frontera hace que los primeros elementos de X e Y, así como los últimos elementos, estén alineados entre sí. En otras palabras, la alineación se atribuye a las secuencias enteras X e Y.

La condición de monotonía refleja el requisito de la sincronización: si un elemento en X precede a otro, esto también debería mantenerse para los elementos correspondientes en Y, y viceversa.

Finalmente, la condición del tamaño de paso expresa una clase de condición de continuidad: no se puede omitir ningún elemento en X e Y y no hay replicaciones en la alineación (en el sentido de que todos los pares de índices contenidos en una alineación p son distintos). La Figura 7 ilustra las tres condiciones.

La primera ilustración a) representa el alineamiento admisible que satisface las tres condiciones declaradas previamente. En la segunda ilustración (b) la condición de frontera es violada. Seguidamente, en la tercera ilustración (c) se viola la condición de monotonía. Finalmente, en la cuarta ilustración (d) la condición del tamaño de paso es violada.

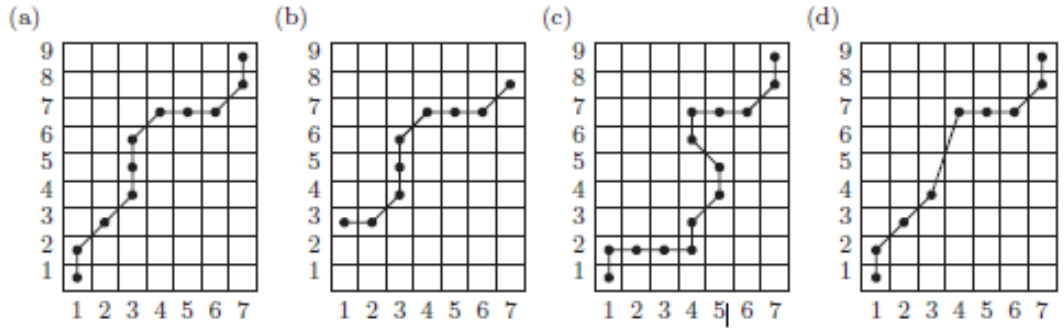

Figura Estado del arte - 7: Alineamiento entre dos secuencias (DTW)
Fuente: Reproducción de [12]

El coste total c_p *(X, Y)* de un camino óptimo *p* entre X e Y con respecto a la medida de coste local c se define como:

$$c_p(X, Y) := \sum_{\ell=1}^{L} c(x_{n_\ell}, y_{m_\ell}).$$

Asimismo, una alineación óptima entre *X* e *Y* es un camino óptimo p^* que tiene un coste total mínimo entre todas las posibles alineaciones. La distancia *DTW (X, Y)* entre *X* e *Y* se define entonces como el coste total de p^*: DTW (*X*, *Y*) = c_{p*} (*X*, *Y*) = min {c_p (*X*, *Y*) donde *p* es un camino (*N*, *M*)}

Caben destacar algunas observaciones sobre la distancia DTW. En primer lugar, tener en cuenta que, aunque la distancia DTW esté bien definida, puede haber varias alineaciones de un coste total mínimo. En segundo lugar, se puede observar fácilmente que la distancia DTW es simétrica en el caso de que la medida de coste local c sea simétrica. Por ejemplo, se obtiene DTW (*X*, *Y*) = 0 para las secuencias *X*: = (x_1, x_2) e *Y*: = (x_1, x_1, x_2, x_2, x_2) en el caso c (x_1, x_1) = c (x_2, x_2) = 0.

Para determinar un camino óptimo *p**, se puede probar con cada alineamiento posible entre *X* e *Y*. Este procedimiento, sin embargo, conduciría a una complejidad computacional que es exponencial en las longitudes *N* y *M*. Por tanto, se utiliza un algoritmo (descrito en la sección siguiente) O (*NM*) que se basa en la programación dinámica. Para ello, definimos las secuencias de prefijos *X (1: n) =* {x_1, x_2, ..., x_N} para n ∈ [1: N] e *Y (1: m) =* {y_1, y_2, ..., y_M} para m ∈ [1: M] y se establece D (*n*, *m*) = DTW (X (1: *n*), *Y* (1: *m*)).

Los valores *D* (n, m) definen una matriz *N* × *M*, que también se denomina matriz de coste acumulado. Esta se encuentra descrita en la sección siguiente.

2.1.3.1.2 Algoritmo Camino Óptimo

Este algoritmo se emplea para hallar el camino optimo p^* entre dos secuencias *X* y *Y*. Este recibe como parámetro de entrada la matriz de coste acumulado y devuelve como salida el camino óptimo p^*

Se define la matriz de coste acumulado *D* de la siguiente manera:

- $D(n, 1) = \sum_{k=1}^{n} c(x_k, y_1)$ para $n \in [1:N]$
- $D(1, m) = \sum_{k=1}^{m} c(x_1, y_k)$ para $m \in [1:M]$
- $D(n, m) = \min\{D(n\text{-}1, m\text{-}1), D(n\text{-}1, m), D(n, m\text{-}1)\} + c(x_n, y_m)$ para $1 < n \leq N$ y para $1 < m \leq M$.

El procedimiento para hallar el camino optimo p^* es el siguiente; el camino optimo $p^* = (p_1, \ldots, p_L)$ se calcula en orden inverso tomando como comienzo $p_L = (N, M)$. Supongamos que $p_\ell = (n, m)$. En el caso de que $(n, m) = (1,1)$, se debe tener que $p_\ell = 1$ ya habríamos terminado. En caso contrario,

$$p_{\ell-1} = \begin{cases} (1, m-1), & \text{si } n = 1 \\ (n-1, 1), & \text{si } m = 1 \\ \textit{argmin}\{D(n\text{-}1, m-1), D(n\text{-}1, m), D(n, m\text{-}1)\} & \text{en caso contrario} \end{cases}$$

donde tomamos el par lexicográficamente más pequeño en caso de que "argmin" no sea un valor único.

A continuación, la figura 8 muestra el camino de un camino óptimo p^* (trazo en color blanco) para las secuencias de la Fig.7. Tenga en cuenta que p^* sólo cubre las celdas de *C* que presentan bajo coste (ver Fig. 7). La matriz de coste acumulada resultante *D* se muestra en la Fig. 7b en la que se muestra el camino óptimo p^*.

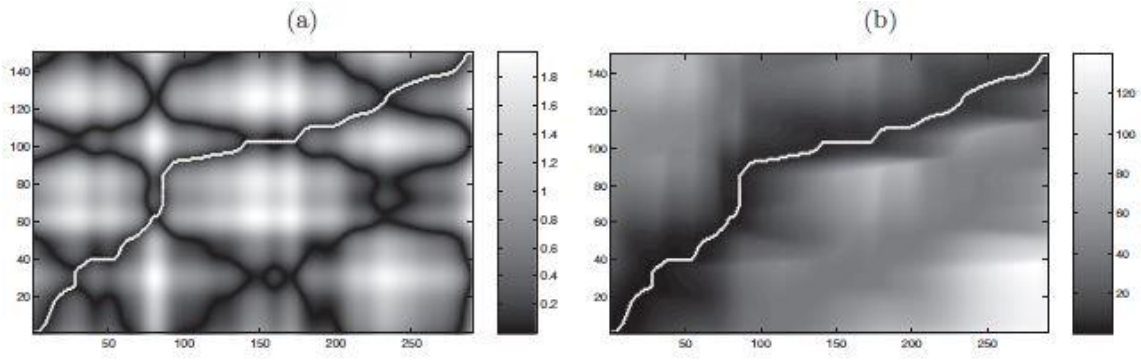

Figura Estado del arte - 8: Representación de matriz de coste (acumulado) (DTW). Fuente: Reproducción de [12]

2.1.3.1.3 S-DTW

En diversas aplicaciones, las secuencias que se quieren comparar muestran una diferencia significativa en cuanto a longitud. En lugar de alinear estas secuencias de manera global, a menudo el objetivo es encontrar una subsecuencia dentro de la secuencia más larga que encaja óptimamente en la secuencia más corta, véase la Fig.9. Por ejemplo, suponiendo que la secuencia más larga representa una base de datos dada y la secuencia más corta representa una consulta, un objetivo típico es identificar el fragmento dentro de la base de datos que es más similar a la consulta.

El problema de encontrar subsecuencias óptimas se puede resolver como una variante del algoritmo descrito anteriormente (alineamiento temporal dinámico), como se describirá en esta sección.

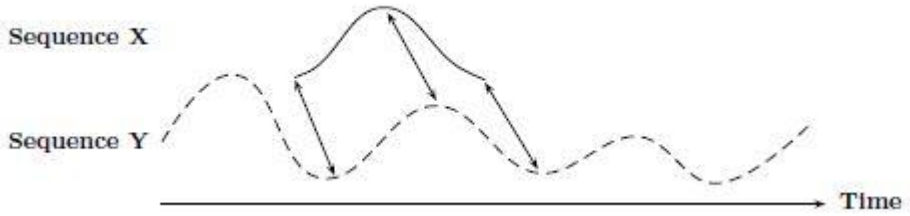

Figura Estado del arte - 9: Alineación óptima en el tiempo de la secuencia X con una subsecuencia de Y (S-DTW).
Fuente: Reproducción del [12]

Dadas las siguientes secuencias de características: $X = \{x_1, x_2, ..., x_N\}$ y $Y = \{y_1, y_2, ..., y_M\}$, se asume que la longitud M es mucho mayor que la longitud N. Seguidamente, fijamos una función de coste local c.

La finalidad de este algoritmo es encontrar una subsecuencia Y $(a^*: b^*)$: = $\{y_{a^*}, y_{a^*+1}, ..., y_{b^*}\}$ con $1 \leq a^* \leq b^* \leq M$ que minimiza la distancia DTW de X sobre todas las posibles subsecuencias de Y. En otras palabras, $(a^*, b^*) = argmin\ (DTW\ (X, Y\ (a: b))$.

Los índices a^* y b^* se pueden calcular mediante una pequeña modificación en la inicialización del algoritmo del camino óptimo (ver sección anterior). La idea básica es no penalizar las omisiones en la alineación entre X e Y que aparecen al principio y al final de Y.

Se modifica la definición de la matriz de coste acumulado D estableciendo $D\ (n, 1)$: $\sum^{n}_{k=1} c\ (x_k, y_1)$ para $n \in [1: N]$ y $D\ (1, m)$: = c (x_1, y_m). Los valores restantes de D se definen de manera recursiva para n perteneciente a [2: N] y m perteneciente a [2: N] del mismo modo que se hacía con el algoritmo DTW estándar. El índice b^* se puede determinar a partir de D por b^* = argmin $D\ (N, b)$ siendo $b \in [1: M]$

Para determinar a^* así como el camino óptimo entre X y la subsecuencia Y $(a^*: b^*)$, aplicamos el algoritmo de camino optimo (ver sección anterior), esta vez, sin embargo, se comienza con $p_L = (N, b^*)$. Sea $p^* = \{p_1, ..., p_L\}$ el camino resultante. Entonces $a^* \in [1: M]$ es el índice máximo tal que $p_\ell = (a^*, 1)$ para algunos $\ell \in [1: L]$. En otras palabras, todos los elementos de Y a la izquierda de y_{a*} y a la derecha de y_{b*} no se consideran en la alineación y no causan ningún coste adicional.

El camino de alineación óptimo entre X e Y $(a^*: b^*)$, viene dado por $(p_\ell, ..., p_L)$. Al mismo tiempo, la subsecuencia óptima Y $(a^*: b^*)$ en general no está definida de una única manera – existen diversas posibilidades para la elección del índice b^* y en el índice a^* se escoge el máximo valor.

Finalmente, se describe cómo la matriz de coste acumulado D se puede usar para derivar una lista completa de subsecuencias de Y que están cerca de X con respecto a la distancia de DTW. Para ello, definimos una función de distancia; $\Delta: [1: M] \rightarrow \mathbb{R}$, $\Delta(b) = D(N, b)$, que asigna a cada índice $b \in [1: M]$, la distancia mínima DTW $\Delta(b)$ que se puede alcanzar entre X y una subsecuencia Y $(a: b)$ de Y que termina en y_b. La complejidad computacional de este algoritmo es O (NM).

2.1.3.2 Modelo oculto de Márkov

El modelo oculto de Márkov (HMM) es un modelo estadístico que es utilizado en diversos campos. Especialmente, destaca en los sistemas de reconocimiento del habla para reconocer secuencias de series de tiempo de los parámetros del discurso como dígitos, caracteres, palabras o frases.

En el reconocimiento del habla, estamos interesados en predecir la palabra pronunciada de una señal de voz grabada. Para este propósito, el reconocedor del habla trata de encontrar la secuencia de fonemas (estados) que dio lugar al sonido pronunciado real (observaciones). Puesto que puede haber una gran variación en la pronunciación real, los fonemas originales (y en última instancia, la palabra proferida) no pueden ser observados directamente, y necesitan ser predichos. [13]

Un modelo oculto de Márkov o HMM (Hidden Márkov Model) es un modelo estadístico en el que se asume que el sistema a modelar es un proceso de Márkov de parámetros desconocidos. El fin es hallar los parámetros desconocidos de esta cadena a partir de los parámetros observables. [14] A diferencia de un modelo de Márkov los estados no son visibles y las salidas observables no se corresponden de forma determinista, sino probabilística, con los estados del sistema. [15]

A continuación, una representación que muestra la topología de este modelo. Como se puede observar, las salidas observables *y* se corresponden de manera probabilística pues cada estado *x* tiene asociada una probabilidad de transición *a*, y de salida *b*.

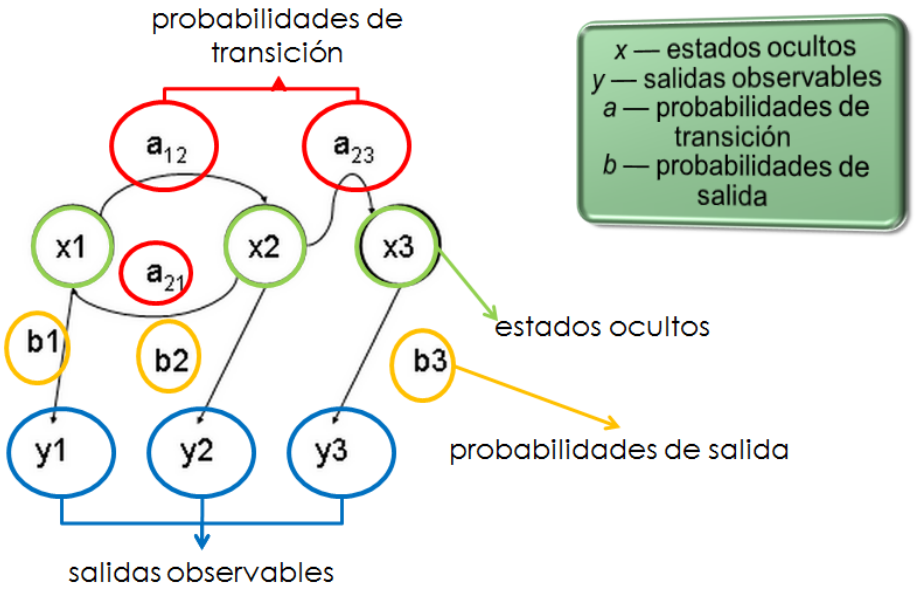

Figura Estado del arte - 10: Topología (Modelo oculto de Márkov). Fuente: [16]

2.1.3.3 Redes Neuronales

Las redes neuronales tienen una larga historia en el reconocimiento del habla, por lo general en combinación con modelos ocultos de Márkov. [17] Han destacado últimamente con las mejoras significativas en el modelado acústico producido por las redes neuronales profundas. [18]

Una red neuronal puede definirse como un modelo de razonamiento basado en el cerebro humano. El cerebro consiste en un conjunto densamente interconectado de células nerviosas, o unidades básicas de procesamiento de información, llamadas neuronas. Mediante el uso de múltiples neuronas simultáneamente, el cerebro puede realizar sus funciones mucho más rápido que las computadoras más rápidas en existencia hoy en día.

Podemos distinguir dos modelos: redes neuronales artificiales (ANN) y redes neuronales profundas (DNN). El primer caso (ANN) se refleja en la Figura 11, y el segundo caso (DNN) en la Figura 12.

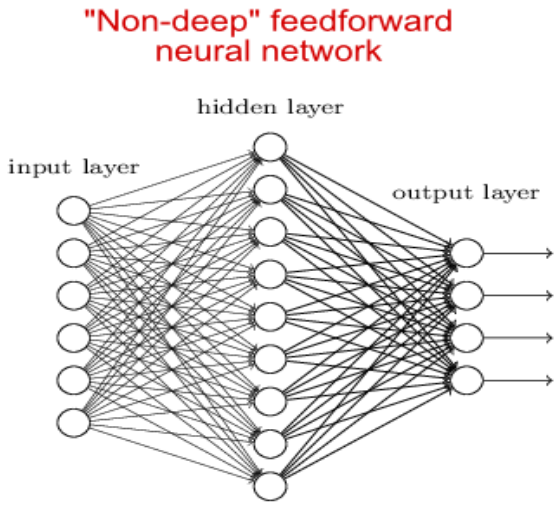

Figura Estado del arte - 11: Topología (ANN).
Fuente: [19]

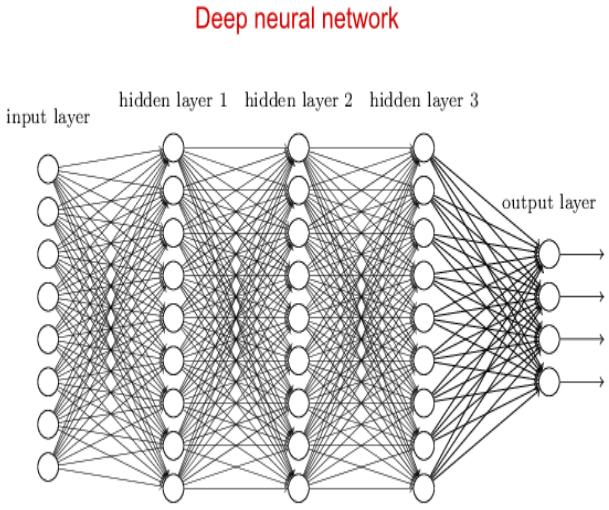

Figura Estado del arte - 12: Topología (DNN).
Fuente: [19]

2.2 Sistemas de búsqueda en voz

Se han realizado investigaciones significativas sobre la recuperación de documentos hablados (SDR), detección de palabras clave (KWS), detección de términos hablados (STD), consulta mediante ejemplo (QbE) o enfoques de consultas habladas para abordar este problema.

Actualmente, existen varios enfoques de cara a implementar un sistema de búsqueda en voz. El método en el que se emplean fonemas es el más rápido para el procesamiento, sobre todo porque el tamaño de la gramática es muy pequeño, con un fonema como unidad básica de reconocimiento. [20] Los sistemas descritos a continuación usan este procedimiento.

2.2.1 Detección de términos hablados (STD)

La detección de términos hablados tiene como objetivo encontrar palabras individuales o secuencias de palabras dentro de archivos de audio. Por lo general, se basa en una entrada basada en texto, comúnmente la palabra o fonema del término a realizar la búsqueda. Por esta razón, STD también se llama STD basada en texto.

Un enfoque común de este sistema es emplear un sistema de reconocimiento de voz continuo de gran vocabulario (LVCSR). Este sistema utiliza un conjunto de palabras como unidad básica. Este enfoque requiere de cientos de miles de palabras a emparejar con el audio. [21]

No obstante, además de este enfoque también STD presenta el enfoque en sub unidades de palabra que busca las representaciones de subunidades de palabra de los términos de búsqueda dentro de la salida de un sistema de reconocimiento de voz de subunidades de palabra.

El enfoque LVCSR típicamente obtiene un mejor rendimiento que el enfoque basado en subunidades de palabra gracias a la información léxica que emplea. Sin embargo, el enfoque basado en subunidades de palabra tiene la ventaja única de que puede detectar términos que consisten en palabras que no están en el vocabulario del reconocedor - términos fuera del vocabulario (OOV) - mientras que el enfoque basado en palabras sólo puede detectarlas en el vocabulario (INV). [22]

2.2.2 Consulta mediante ejemplo (QbE)

La gestión de bases de datos multimedia cada vez más extensas requiere bastante tiempo cuando se hace de manera totalmente manual. En consecuencia, los sistemas automáticos son necesarios para aligerar la gestión. La consulta mediante ejemplo tiene como propósito la recuperación automática de muestras de una base de datos, que son similares al ejemplo proporcionado por el usuario.

La consulta mediante ejemplo se puede definir como 'un método para buscar un ejemplo de un objeto o una parte de él en otros objetos'. Esto ha sido ampliamente utilizado en aplicaciones de audio como la clasificación de sonido, recuperación de información de música y la recuperación de documentos hablados. [23] Frente a sistemas de búsqueda en voz en los que la consulta está en forma textual (por ejemplo, el STD) tienen la ventaja de que los sistemas QbE pueden hacerse independientes del idioma, lo que facilita mucho su utilización.

La consulta mediante ejemplo difiere de la clasificación supervisada dado que no hay clases predefinidas para las que el sistema pueda ser entrenado. Por lo tanto, a menos que la similitud se defina de antemano de acuerdo con un cierto criterio, la consulta empleando sólo un ejemplo no sería un problema bien definido.

Este sistema se suele desarrollar de la siguiente manera [24]: en primer lugar, las características se extraen del ejemplo (ver sección 2.1.2) y todas las muestras en la base de datos. Seguidamente, las distancias entre los vectores de características del ejemplo y las muestras de base de datos se calculan utilizando una determinada métrica de distancia. Este segundo paso se suele realizar mediante un algoritmo (ver sección 2.1.3). Finalmente, se recuperan las muestras de bases de datos que tienen la distancia más corta al ejemplo.

2.2.2.1 Usos en QbE

A parte del uso en el que se ha centrado este TFG, existen otras utilidades del sistema QbE, como pueden ser las descritas a continuación.

2.2.2.1.1 Consulta de imágenes mediante ejemplo

Actualmente, millones de personas buscan imágenes en Internet, tanto por motivos profesionales como personales. Sin embargo, cada vez el número de imágenes en la red aumenta y estas no están correctamente etiquetadas por ende se hace más difícil encontrarlas. Esto da lugar al desarrollo de métodos para asignar términos a las imágenes que no tienen etiquetas asignadas por el usuario, ni una descripción textual (Duygulu et al., 2002; Lavrenko et al., 2003; Guillaumin and Mensink, 2009).

Estos métodos aprenden a asociar la presencia y ausencia de etiquetas con las características visuales de una imagen, como las distribuciones de color y textura, la forma y los puntos de interés, y pueden generar automáticamente una agrupación de términos para una imagen sin etiquetar. [25]

El sistema de consultas de imágenes mediante ejemplo o *Content-Based Image Retrieval* (CBIR) emplean los métodos descritos anteriormente para la recuperación de imágenes. Este sistema tiene su origen en 1992, este término fue usado por primera vez por T. Kato para describir experimentos de recuperación automática de imágenes de una base de datos, en base a los colores y formas presentes. [26]

El primer sistema CBIR fue desarrollado por IBM y se llamó QBIC (*Query by Image Content*). [27] Este sistema recibe como parámetro de entrada una imagen, se extraen las características visuales y se comparan con el conjunto de características que se obtienen de una base de datos de imágenes de esta manera se obtienen las imágenes recuperadas. El proceso mencionado previamente se refleja en la Figura 13.

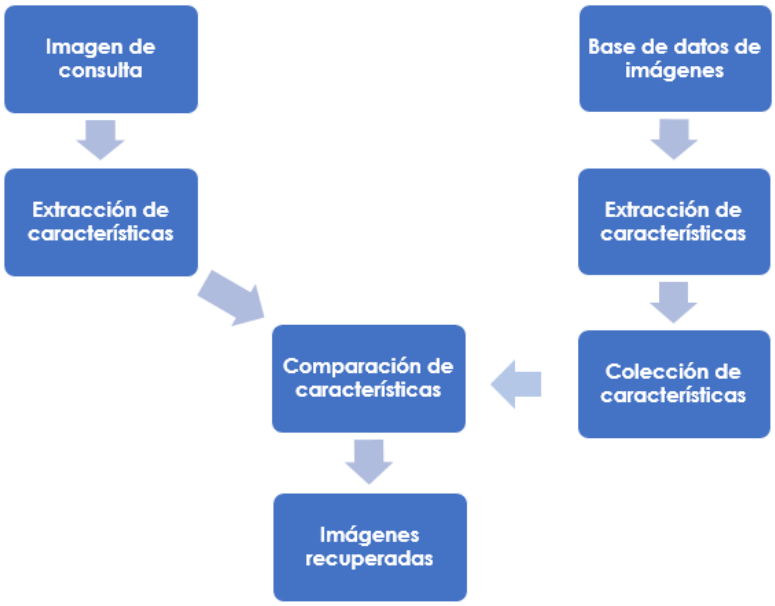

Figura Estado del arte - 13: Sistema CBIR.

Un sistema del tipo CBIR, es el sistema: *Color-Based Image Retrieval* que trata de recuperar imágenes basándose en la distribución de color de las imágenes. Para recuperar imágenes con similar distribución de color son necesarios dos pasos. El sistema explicado a continuación es un breve resumen de esta fuente: [28]

El primer paso, sería calcular un histograma 3D de la imagen de consulta.

El segundo paso sería una vez calculado el histograma, se ejecuta el algoritmo de búsqueda. El algoritmo de búsqueda tras cargar el histograma, se extraen las características estas se calculan usando la distancia entre la imagen de la consulta y las imágenes de la base de datos de imágenes, se escogerían las imágenes que tienen el menor valor (una distancia pequeña)

Finalmente, tras ejecutar los programas correspondientes la salida se muestra en la figura 14 donde se muestra la imagen de consulta junto con las (en este caso son 11) imágenes más próximas. [28]

Figura Estado del arte - 14: Salida del algoritmo (*Color-Based Image Retrieval*)
Fuente: [28]

2.2.2.1.2 Recuperación de información musical

En la era digital, una gran cantidad de datos de audio se producen y se consumen todos los días en una gran variedad de idiomas. Pueden estar en forma de música, noticias de televisión, conferencias en el aula, libros de audio, podcasts, archivos de centros de llamadas e incluso grabaciones personales de audio. Con este crecimiento exponencial de contenido multimedia digital, la búsqueda de audio se vuelve esencial para la recuperación rápida de información de archivos de audio.

Una técnica prometedora para recuperar la información musical o *Music Information Retrieval* (MIR) es empleando el sistema de recuperación de búsqueda mediante ejemplo, donde un usuario puede recibir una lista de piezas musicales clasificadas por su similitud con una pieza musical (ejemplo) que el usuario da como una consulta. [29]

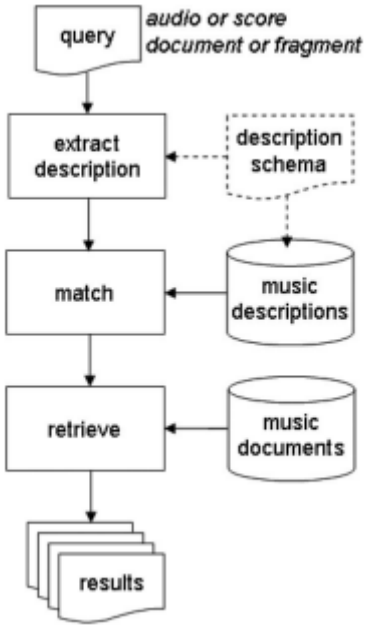

Figura Estado del arte - 15: Recuperación de información musical (*MIR*)
Fuente: [30]

En la recuperación de información musical existen varios sistemas, como puede ser *Query-by-Humming (QbH)*. Este sistema utiliza como ejemplo un archivo de audio en el que se tararea la melodía de una canción y este se busca en una base de datos musical. Tal sistema sería útil en cualquier base de datos multimedia que contiene datos musicales proporcionando una forma alternativa y natural de consulta. También se puede imaginar un uso generalizado de tal sistema: en la industria de la música comercial, radio musical y cadenas de televisión, tiendas de música e incluso para el uso personal. [31]

En la actualidad, es notable la gran cantidad de dispositivos que son capaces de reproducir archivos multimedia este hecho se puede relacionar con el desarrollo de motores comerciales de búsqueda musical de consulta mediante ejemplo (QbE) p. ej. FreeDB [32] y MusicBrainz [33] que son dos sistemas basados en la recuperación musical que usan la QbE disponibles en línea. Otro sistema popular por el uso de la QbE para la recuperación musical son Shazam [34] está disponible para teléfonos móviles como aplicación. Este sistema emplea métodos de huellas dactilares acústicas para buscar una canción que el usuario este escuchando en la radio o en su entorno. Además, dispone de una base de datos de más 11 millones de canciones. Neuros [35] es otro sistema de consulta mediante ejemplo que permite al usuario conectarse a un servicio en línea para buscar la canción que desee con un clip de 30 segundos de ella.

3 Diseño

El diseño puede dividirse en varias fases que fueron necesarias para llevar a cabo el desarrollo del sistema.

3.1 Material

El material que ha sido empleado para el desarrollo de este sistema es el mismo que fue proporcionado a los participantes de la evaluación competitiva ALBAYZIN (noviembre 2016). En dicha evaluación se emplean dos bases de datos diferentes. Estas bases de datos son EPIC y MAVIR. Para la base de datos MAVIR, se proporcionaron a los participantes tres conjuntos de datos separados (es decir, para entrenamiento, desarrollo y prueba). Para la base de datos EPIC, sólo se proporcionaron datos de prueba. Para este sistema únicamente se trataron los datos de prueba (*test*) de ambas bases de datos. La base de datos de MAVIR tiene una duración de aproximadamente 2 horas en total, extraída de 3 ficheros de audio. La base de datos EPIC contiene datos de los discursos del Parlamento Europeo registrados en 2004 en inglés, español e italiano, junto con sus correspondientes interpretaciones simultáneas a los otros idiomas. Sólo los discursos originales en español se utilizan para la evaluación. [1]

3.2 Pre-procesamiento

La primera fase es la de pre-procesamiento, detallada posteriormente en la sección de desarrollo. En esta fase se emplea como material ambas bases de datos mencionadas anteriormente y el uso de *scripts* auxiliares que fueron implementados con el propósito de pre-procesar los datos. En tal caso, esta fase consiste fundamentalmente en pre-procesar todos los archivos de audio y extraer sus características. Las características que hemos seleccionado para el TFG son las probabilidades a posteriori de los fonemas, por sus buenos resultados previos en aplicaciones similares [36]. Por tanto, para pre-procesar estos archivos es necesario el uso de un reconocedor de fonemas. Este reconocedor de fonemas (Brno) usa varios decodificadores fonéticos. Los decodificadores fonéticos checos (CZ), ingleses (EN), húngaros (HU) y rusos (RU) desarrollados por la Universidad Tecnológica de Brno fueron utilizados para obtener probabilidades a posteriori. Conjuntamente, es necesario realizar diversas configuraciones explicadas con detalle posteriormente y el uso de la herramienta HTK.

3.3 Algoritmo de comparación

Por último, esta fase consiste en la implementación de una variante del algoritmo de alineamiento temporal dinámico. Esta variante es el S-DTW, explicado de forma detallada en el estado del arte. Esta fase, nos proporciona los límites de una posible detección de la consulta dentro de la base de datos y una puntación para esta detección. Después de eso, se establece un umbral basado en las puntuaciones de todas las posibles detecciones. Este paso es imprescindible para aceptar únicamente las detecciones con las coincidencias más altas.

4 Desarrollo

4.1 Pre-procesamiento

Como punto de partida, se nos proporcionan unos archivos de audio (de extensión WAV) los cuales van a ser empleados para realizar consultas (*queries*) en los archivos de audio mayor duración. Estos ficheros provienen de dos bases de datos distintas (EPIC y MAVIR). No obstante, este hecho no influye en el tratamiento de los ficheros, pues se procesan ambos ficheros mediante el mismo procedimiento.

Previamente, es necesario elaborar unos scripts auxiliares para poder modificar la frecuencia de muestreo. La frecuencia de muestreo de todos los ficheros de audio proporcionados es de 16000 kHz y para poder usar el reconocedor, la frecuencia debía ser de 8000 kHz. Esta conversión es posible utilizando el programa SOX. A continuación, se muestra la salida después de transformar un fichero de audio usando el programa SOX:

Input File: 'TEST-0000.wav'
Chaneles: 1
Sample Rate: 8000
Precision: 16-bit
Duration: 00:00:00.69 = 11038 samples ~ 51.7406 CDDA sectors
File Size: 22.1k
Bit Rate: 257k
Sample Encoding: 16-bit Signed Integer PCM

Figura Desarrollo* - 16*: Salida programa SOX.

Seguidamente, se usa el reconocedor desarrollado por la Universidad Tecnológica de Brno [37] que contiene cuatro decodificadores de fonemas: checo, ruso, inglés y húngaro para la extracción de características.

Para ello se realiza un *script* y haciendo uso de las herramientas que se proporcionan para dicho reconocedor obtenemos como salida un fichero de tipo post (probabilidades a posteriori). Los decodificadores de este reconocedor para cada fonema tienen tres estados diferentes y se emite una probabilidad a posteriori para cada uno de ellos.

El fichero de salida obtenido tras usar el reconocedor está en formato binario. Por consiguiente, es necesario el uso de la herramienta HTK para poder transformarlo a formato ASCII. Dadas las condiciones que anteceden, será necesario un script auxiliar que realice este proceso. Así pues, haciendo uso del programa HTK, concretamente del programa HList el fichero en binario se transforma a fichero en ASCII. En definitiva; el fichero resultante es un fichero de texto en el cual se muestran los vectores que contienen para cada estado su probabilidad a posteriori.

Con el fin de reducir el vector por cada marco se suman las probabilidades a posteriori de los 3 estados de cada fonema. Por otra parte, el fichero tiene una longitud de (N+1), y sabiendo que cada fonema tiene 3 estados y estos están separados una longitud N se suman los 3 estados por fonema. Esta operación se realiza en un *script* auxiliar, que deja el fichero con el mismo número de líneas y N+1 probabilidades.

Finalmente, en el fichero resultante se muestran 3*(N+1) probabilidades a posteriori (*posteriors*) donde N es el número de fonemas del reconocedor en el idioma que se maneje y el +1 es porque también hay un modelo de silencio. [38] Este fichero se obtiene tanto para las consultas como para las bases de datos, este fichero es el que se usaría posteriormente en el algoritmo de comparación como parámetro de entrada.

4.2 Algoritmo de comparación

Para el desarrollo de un sistema de búsquedas mediante ejemplo es necesario la implementación de un algoritmo que pueda comparar los ejemplos y las bases de datos tras ser pre-procesadas. Tal y como se enunció en el apartado anterior (ver sección 3.3) el algoritmo implantando es una variante del alineamiento temporal dinámico, este es el *Subsequence* DTW (S-DTW) [11] explicado de manera detallada en el Estado del Arte (ver sección 2.1.3.1.3). El algoritmo fue programado en Python.

Para llevar a cabo el algoritmo S-DTW se definen los parámetros de entrada y de salida de dicho algoritmo. Por un lado, tiene como parámetros de entrada una consulta y una base de datos, estos son los ficheros finales obtenidos en la etapa anterior. Por otro, tiene como parámetro de salida el instante de inicio, fin y coste de la detección del ejemplo en la base de datos.

Son necesarios diversos pasos para realizar el algoritmo. Inicialmente, se define una matriz de coste $M \in \Re$ de tamaño n × m, siendo n el tamaño de la consulta y m el tamaño del documento.

$$M_{i,j} = \begin{cases} c\,(q_i, d_j) & \text{en el caso de que } i = 0 \\ c\,(q_i, d_j) + M_{i-1,0} & \text{en el caso de que } i > 0,\ j = 0 \\ c\,(q_i, d_j) + M^*(i, j) & \text{en caso contrario} \end{cases}$$

donde c (q_i, d_j) es una función que define el coste entre el vector de consulta q_i y el vector del documento d_j, y

$$M^*(i, j) = min\,(M_{i-1,\,j}, M_{i-1,\,j-1}, M_{i,\,j-1}).\ [36]$$

La función de coste se define como el coeficiente de la correlación de *Pearson*. Para hallar el coeficiente, uso la función de la librería *scipy* de Python llamada *pearsonr* que devuelve *r* y *p* siendo *r* el coeficiente de correlación de Pearson y *p* el indicador del grado de significancia estadística del mismo [39].

Al ser este valor (*r*) un valor negativo se redefine la función de coste $c\ (q_i, d_j)$ de la siguiente manera [36]:

$$c(q_i, d_j) = \frac{1 + r(q_i, d_j)}{2}.$$

Llegados a este punto, ya tendríamos definitivamente definida la matriz de coste de nuestro algoritmo. El siguiente paso a realizar es hallar el camino óptimo que es el alineamiento entre la consulta y la base de datos. Este camino está delimitado por los índices a^* y b^* (definidos anteriormente en la sección 2.1.3.1.3), siendo b^* el final del mejor camino optimo y el índice a^* el inicio del camino.

Para hallar el índice b^*, según viene definido [11] por esta fórmula: b^* = argmin M (n, b) siendo $b \in [1: m]$. La matriz de coste se almacenan los mayores costes en la última fila, y en la primera fila, consecuentemente, debería de tener el menor coste posible. El valor de la matriz de coste en el índice *b** devolvería el *score* (puntación, el coste mínimo) que sería uno de los parámetros de salida que retorna el algoritmo para una posterior evaluación.

Después de haber hallado el índice b^*, para hallar el índice *a** es necesario usar el algoritmo de camino optimo (ver sección 2.1.3.1.2) y recorrer inversamente la matriz partiendo desde el índice b^*. El camino optimo viene dado por $p^* = (p_1, \ldots, p_L)$, que contiene los índices de la matriz hasta llegar a la primera fila de la matriz. Se comienza con $p_L = (N, b^*)$ desde este se recorre la matriz de la siguiente manera [11]:

.

$$p_{L-1} = \begin{cases} (1, m-1), & \text{si n = 1} \\ (n-1, 1), & \text{si m = 1} \\ \textit{argmin}\ \{M(n\text{-}1, m-1), M(n\text{-}1, m), M(n, m-1)\} & \text{en caso contrario} \end{cases}$$

En este caso n es la longitud de la consulta, y m es la longitud del documento.

Si $L = 1$ significa que ya habríamos terminado de recorrer la matriz y por tanto si devolvemos el primer elemento de p^*, se obtiene de esta manera el índice a^*.

En suma, tendríamos el índice a^*, el índice b^* y el coste mínimo de la consulta, se podría dar por finalizado el algoritmo en este instante. Sin embargo, es posible que una consulta aparezca varias veces en un documento, especialmente si es una grabación de larga duración.

Por lo tanto, se realizan una serie de mejoras descritas a continuación. El coste se normaliza por la longitud de la consulta dado que el coste se acumula trama a trama y con una consulta larga se obtiene un coste alto y con una consulta corta un coste bajo. La solución a este problema se resuelve dividiendo el coste por el valor de *n* (longitud de la consulta) y se logra de esta manera el coste medio por trama que siempre se encuentra entre 0 y 1. Cuando el coste medio este próximo a 0 indica que los marcos de la consulta tienen mucha correlación con los marcos del documento y en caso contrario (cuando este próximo a 1) indica que no tiene correlación.

De esta manera, tras haber obtenido el coste por trama, como se ha explicado anteriormente se halla el b^* mínimo y se calcula a^*. En este caso se realiza otra modificación. Con lo anterior, ya se habría terminado el programa y hubiera devuelto los parámetros de salida. Para detectar si realmente la consulta se encuentra en el documento se hace uso de un umbral. Este umbral inicialmente es de 0.25. Si tras haber hallado el coste medio y los índices, el coste es mayor al umbral se termina de buscar en ese documento y se termina el programa. Si por lo contrario el coste es menor al umbral devolvemos inicio, final y coste promedio. Además, en la posición correspondiente al índice b^* introducimos un coste altísimo y se vuelve al paso de buscar el b^* mínimo. Así, se encuentran todos los caminos que tengan un coste por trama inferior al umbral.

Finalmente, tras realizar estas mejoras los caminos que se encuentran suelen ser próximo al primer índice b^* hallado, es decir caminos que empiezan en b^*+1, en b^*+2, en b^*-1 y sucesivamente. Como último perfeccionamiento se introduce un margen de exclusión de +/- *P* marcos, donde *P* tiene un valor suficientemente alto para que no se detecten consultas en la misma zona que se detectó ya una. Para implementar esta mejora, se introduce un coste elevado no solo en la posición b^* si no en las posiciones comprendidas entre b^*+/- P. El algoritmo solo encuentra caminos suficientemente separados.

5 Pruebas y resultados

5.1 Pruebas

Una vez concluida la fase de implementación del algoritmo, se procede a comprobar que este funciona correctamente. Se realiza una prueba inicial de recortar un fragmento de audio de un documento para comprobar que lo encuentra. Con ello comprobamos que detecta el audio en el segundo exacto en el que comienza el fragmento de audio en el documento.

No obstante, este no es el cometido del sistema si no el de encontrar archivos de audio que no tenemos consciencia de que se encuentren en los documentos. Tras realizar varias pruebas, usando distintos ficheros de consultas y documentos de la base de datos EPIC de la carpeta de test se ha podido comprobar que el algoritmo localiza correctamente una consulta en el documento.

A continuación, se muestran las salidas del algoritmo en diferentes ocasiones (con diferentes ficheros de consulta y documento) en los cuales se detecta o no de manera satisfactoria.

- Caso 1: En este caso la consulta se encuentra correctamente. La consulta es el archivo de audio "TEST-0022.wav", y el documento en el que se busca la consulta es el archivo de audio "12-02-04-m-010-org-es_8k.wav". La matriz de coste es de 60x14292, la puntuación normalizada es: 0.27011440017. Estableciendo un umbral de 0.5 se halla la consulta en el documento empezando en el segundo 136.38 (índice a^*) y termina en el segundo 136.61 (índice b^*).

- Caso 2: En este caso la consulta no se encuentra en el documento y el algoritmo detecta que no está en el documento. La consulta es el archivo de audio "TEST-0140.wav" y el documento en el que se busca la consulta es el archivo de audio "12-02-04-m-028-org-es_8k.wav". La matriz de coste es de 68x2780, la puntuación normalizada supera el umbral. Estableciendo un umbral de 0.31, el algoritmo dictamina que la consulta no se encuentra en el documento. Claramente el establecimiento del umbral va a ser un punto crítico en este sistema.

- Caso 3: En este caso la consulta no se encuentra en el documento y el algoritmo detecta erróneamente que está en el documento. La consulta es el archivo de audio "TEST-0064.wav" y el documento en el que se busca la consulta es el archivo de audio "10-02-04-m-058-org-es_8k.wav". La matriz de coste es de 69x14891, la puntuación normalizada es: 0.252746084462. Estableciendo un umbral de 0.30, se halla la consulta en el documento empezando en el segundo 147.48 (índice a^*) y termina en el segundo 147.71 (índice b^*). En este caso el umbral debería ser más restrictivo, pero si lo hacemos tan restrictivo como para que rechace esta detección no detectara el Caso 1.

- Caso 4: En este caso la consulta se encuentra en el documento y el algoritmo no lo detecta. La consulta es el archivo de audio “TEST-0173.wav” y el documento en el que se busca la consulta es el archivo de audio “11-02-04-m-017-org-es_8k.wav”. La matriz de coste es de 75x20831, la puntuación normalizada supera el umbral. Estableciendo un umbral de 0.275, el algoritmo dictamina que la consulta no se encuentra en el documento. Este caso de nuevo requeriría un umbral inferior.

5.2 Resultados

Después de las primeras pruebas iniciales, se realiza una evaluación objetiva para obtener unos datos significativos de la eficiencia del algoritmo implementado. Para ello se han realizado varias búsquedas con cinco consultas diferentes en los 21 archivos (base de datos de EPIC) que son denominados documentos. Se realizan estas búsquedas variando *n*: el número de veces que se realiza el paso de encontrar el coste mínimo en la matriz de coste (ver sección 4.2) además de variar el umbral para detectar mayor o menor cantidad de consultas.

Se calculan diferentes probabilidades como son la de falsa aceptación (%FA) la cual indica en porcentaje la cantidad de veces que se detecta “algo” en la base de datos (por cada segundo de audio contenido en la base de datos), pero es diferente a la consulta que debería encontrarse en la base de datos como se puede ver en el caso 3 (ver sección anterior). Otra probabilidad que es calculada es la de falso rechazo (%FR) que indica cuando el algoritmo decide que no está la consulta en el documento, pero ciertamente si está como en el caso 4 (ver sección anterior).

Las cinco consultas escogidas son las descritas en la Tabla 2 y estas fueron seleccionadas de manera totalmente aleatoria. Los resultados que se muestran son las palabras correctamente detectadas y las probabilidades de falsa aceptación y falso rechazo expresadas como en la Tabla 1.

Leyenda	
FR	Probabilidad de falso rechazo en %
FA	Probabilidad de falsa aceptación en %

Tabla 1 – Leyenda de la tabla de resultados.

A continuación, en la tabla siguiente se pueden observar los resultados obtenidos tras realizar la búsqueda de cinco consultas en todos los documentos con diferentes umbrales y diferente *n*:

		RESULTADOS			
CONDICIONES	Umbral	0,5	0,31	0,31	0,275
	n	2	5	11	11
QUERIES	Test 0022	2 palabras	1 palabra	2 palabras	0 palabras
		FA = 0,3% FR = 75%	FA = 0,3%, FR = 87,5%	FA = 0,3% FR = 75%	FA = 0,02% FR = 100%
	Test 0046	3 palabras	2 palabras	1 palabra	0 palabras
		FA = 0,3% FR =66,67%	FA = 0,28% FR = 77,77%	FA = 0,29%, FR = 88,9%	FA= 0,07%, FR =100%
	Test 0064	8 palabras	2 palabras	3 palabra	2 palabra
		FA = 0,21% FR = 38,64%	FA = 0,3% FR = 84,26%	FA =0,29% FR = 69,23%	FA = 0,27% FR = 84,26%
	Test 0140	3 palabras	2 palabras	1 palabra	0 palabras
		FA = 0,3% FR = 66,67%	FA = 0,2%, FR = 77,77%	FA = 0,2%, FR = 88,9%	FA = 0,03%, FR = 100%
	Test 0173	2 palabras	2 palabras	2 palabras	2 palabras
		FA = 0,32% FR = 66,67%	FA = 0,32% FR = 66,67%	FA = 0,28% FR = 66,67%	FA = 0,1%, FR = 66,67%

Tabla 2 – Resultados.

El audio *TEST-0022* está en 8 de 21 archivos, El audio *TEST-0046* está en 9 archivos, el audio *TEST-0064* está en 13 y el *TEST-0173* está en 6. Tras contrastar estos dados con los de la tabla 2, podemos concluir que los mejores resultados se obtienen con umbral de 0.5.

No obstante, con este umbral se obtienen las probabilidades de falsa aceptación más altas, esto es consecuencia de tener un umbral relativamente alto aceptando más detecciones de las que realmente hay. Con un umbral menor se alcanzan menores probabilidades de falsa aceptación a la vez que aumentan las probabilidades de falso rechazo. Al aumentar las veces que se realiza el paso de calcular el coste mínimo, no aumenta el número de palabras encontradas. Sin embargo, si se encuentran en diferentes documentos en los que con una *n* menor no se habían encontrado y dejan de detectarse las palabras que fueron detectadas anteriormente.

En determinados casos es mejor establecer un n pequeño y un umbral alto como en la búsqueda de la consulta *TEST-0064* en el que se logran encontrar 8 ocurrencias de 13, es decir, un 61% de eficiencia. Por otro lado, en la búsqueda de la consulta *TEST-0173* en todos los casos se obtienen 2 ocurrencias de 6 en todos los casos, siendo el mejor de los casos un umbral 0.31 y un $n = 11$.

Después de este análisis podemos concluir que los mejores resultados a pesar de tener una probabilidad alta en comparación con los diferentes casos descritos son obtenidos con n = 2 y un umbral de 0.5 pues se hallan la mayor cantidad de palabras.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este TFG se ha cumplido con el objetivo marcado: el desarrollo de un sistema de búsqueda mediante ejemplo. El sistema se ha evaluado con datos de la evaluación competitiva ALBAYZIN 2016 *Search On Speech Evaluation* [1], es decir, se han utilizado los mismos recursos que fueron entregados para dicha evaluación.

El sistema a desarrollar podía ser tratado de diferentes maneras, pues existen diversas maneras de extraer las características de una señal de voz y a su vez diferentes algoritmos de comparación con los que comparar estas características. Sin embargo, las técnicas empleadas para la implementación del sistema son las siguientes: probabilidades a posteriori y la variante S-DTW [12] del algoritmo alineamiento temporal dinámico. La elección de este sistema no es algo arbitrario, este sistema es el que mejor resultados presento la evaluación.

Se ha logrado un funcionamiento muy aceptable del algoritmo de comparación incluyendo mejoras en las que se ha podido comprobar si el sistema que se consultaba estaba en la base de datos o no y escoger el mejor resultado (mejor camino) de detección posible. Estableciendo un umbral mayor o menor, se realizaban un mayor número de detecciones como se ha podido demostrar en la sección anterior.

6.2 Trabajo futuro

Se han identificado un par de mejoras que se podrían incluir en el sistema, que en este sistema no se han podido incluir por motivos de tiempo y complejidad.

El sistema no dispone de una evaluación objetiva como las que se exigían para la evaluación, en las que se utilizaba una métrica ATWV [40]. Esta métrica es empleada tras haber buscado todas las consultas en las bases de datos. El incluir esta evaluación permitiría realizar una comparativa completamente justa con los sistemas enviados a la evaluación.

Al mismo tiempo, se podrían realizar mejoras en otras etapas como puede ser la extracción de características pues si se tomaran logaritmos de las probabilidades a posteriori se podría realizar una mejor detección en el algoritmo.

7. Referencias

[1] Tejedor J. y Toledano D., “The Albayzin 2016 Search On Speech Evaluation Plan”, noviembre 2016, pp 1-3.

[2] Juang, B. H.; Rabiner, Lawrence R. "Automatic speech recognition–a brief history of the technology development”. pp- 6. Retrieved 17 January 2015.

[3] Benesty, Jacob; Sondhi, M. M.; Huang, Yiteng (2008). Springer Handbook of Speech Processing. Springer Science & Business Media. ISBN 3540491252.

[4] First-Hand: The Hidden Markov Model - Engineering and Technology History Wiki: http://ethw.org/First-Hand:The_Hidden_Markov_Model

[5] IBM100 - Pioneering Speech Recognition
http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/speechreco/

[6] H. Bourlard and N. Morgan, Connectionist Speech Recognition—A Hybrid Approach. Norwell, MA: Kluwer, 1994.

[7] P. Schwarz, "Phoneme Recognition based on Long Temporal Context, PhD Thesis", Brno University of Technology, 2009

[8] Hermansky, H., Ellis, D.P.W., and Sharma, S., “Connectionist Feature Extraction for Conventional HMM Systems”, Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, Istanbul, Turkey, 2000

[9] Gaudard, C., Aradilla, G., & Bourlard, H. (2007). Speech Recognition based on Template Matching and Phone Posterior Probabilities (No. LIDIAP-REPORT-2007-006). IDIAP.

[10] http://www.scielo.org.mx/pdf/rmib/v34n2/v34n2a2.pdf

[11] http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs

[12] Müller, M.: Information Retrieval for Music and Motion. Springer-Verlag (2007) PP: 70-82

[13] Rabiner L R. A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE. 1989; PP: 77:257–286.

[14] Modelo oculto de Márkov
https://es.wikipedia.org/wiki/Modelo_oculto_de_M%C3%A1rkov

[15] http://arantxa.ii.uam.es/~jortega/HMMs_ASAL.pdf

[16] Modelos Estocásticos:
http://www.marcoregalia.com/STUFF/UDISTRITAL/Bioinformatica/Actividades/Resumenes%20Clases/modelosEstocasticos.html

[17] H.A. Bourlard and N. Morgan, Connnectionist Speech Recognition: A Hybrid Approach, Kluwer Academic Publishers, 1994

[18] G. Hinton, Li Deng, Dong Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," Signal Processing Magazine, IEEE, vol. 29, no. 6, pp. 82 –97, nov. 2012.

[19] https://i.stack.imgur.com/OH3gI.png

[20] Meteer M., The Right Technology for your Speech Analytics Project". Call Miner. Retrieved .30 September 2016, pp 1-3.

[21] D. Can, E. Cooper, A. Sethy, C. White, B. Ramabhadran, and M. Saraclar, "Effect of pronounciations on OOV queries in spoken term detection," Acoustics, Speech, and Signal Processing, IEEE International Conference on, vol. 0, pp. 3957–3960, 2009.

[22] TK Chia, KC Sim, H Li, HT Ng, in A lattice-based approach to query-by-example spoken document retrieval. Proc. of ACM SIGIR (ACM, New York, USA, 2008), pp. 363–370.

[23] I Szoke, M Fapso, M Karafiat, L Burget, F Grezl, P Schwarz, O Glembek, P Mat˘ejka, S Kontar, J Cernocky, in Proc. of NIST Spoken Term Detection Evaluation Workshop (STD'06). BUT system for NIST – English (NIST, Gaithersburg, MD, USA, 2006), pp. 1–15

[24] M. Helen and T. Lahti, "Query by Example Methods for Audio ´ Signals," in Proc. 7th IEEE Nordic Signal Processing Symposium, Reykjavik, Iceland, June 2006, pp. 302–305

[25] http://homepages.inf.ed.ac.uk/keller/papers/coling14.pdf

[26] Eakins, John; Graham, Margaret. "Content-based Image Retrieval". University of Northumbria at Newcastle. Retrieved 2014-03-10.

[27] Rui, Yong; Huang, Thomas S.; Chang, Shih-Fu (1999). "Image Retrieval: Current Techniques, Promising Directions, and Open Issues". Journal of Visual Communication and Image Representation. 10: 39–62. doi:10.1006/jvci.1999.0413. Retrieved 2 July 2016.

[28] https://ece.uwaterloo.ca/~nnikvand/Coderep/imQuery/documentation.html

[29] Itoyama, Katsutoshi, et al. "Query-by-example music information retrieval by score-informed source separation and remixing technologies." EURASIP journal on Advances in Signal Processing 2010.1 (2011): 172961. pp:1-2.

[30] Casey, Michael A., et al. "Content-based music information retrieval: Current directions and future challenges." Proceedings of the IEEE 96.4 (2008): 668-696.

[31] Ghias, Asif, et al. "Query by humming: musical information retrieval in an audio database." Proceedings of the third ACM international conference on Multimedia. ACM, 1995.

[32] A. Schröder, and M. Keith. "Free database," http://www.freedb.org.

[33] R. Kaye. "the Open Music Encyclopedia," https://musicbrainz.org.

[34] C. Barton, P. Inghelbrecht, A. Wang, and D. Mukherjee. "Shazam Company," http://www.shazam.com/company.

[35] J. Born. "Neuros "; www.neurostechnology.com.

[36] Lopez-Otero, P., Docio-Fernandez, L., Garcia-Mateo, C. GTM-UVigo Systems for Albayzin 2016 Search on Speech Evaluation. Noviembre 2016.

[37] T. J. Hazen, W. Shen, and C. White. Query-by-example spoken term detection using phonetic posteriorgram templates. In Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on, pages 421–426. IEEE, 2009.

[38] Y. Zhang and J. R. Glass. Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams. In Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on, pages 398–403. IEEE, 2009.

[39] https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.pearsonr.html

[40] https://www.nist.gov/sites/default/files/documents/itl/iad/mig/KWS14-evalplan-v11.pdf

Glosario

STD	Spoken Term Detection
QbE	Query by Example
DTW	Dynamic Time Warping
HMM	Hidden Márkov Model
MFCC	Mel Frequency Cepstral Coefficients
PLP	Perceptual Linear Predictive
LP	Linear Prediction
DTF	Discrete Fourier Transform
ITF	Inverse Fourier Transform
ASR	Automatic Speech Recognition
SDR	Spoken Document Retrieval
KWS	Key Word Spotting
OOV	Out Of Vocabulary
INV	In-Vocabulary
ANN	Artificial Neural Network
DNN	Deep Neural Network
QbH	Query by Humming
MIR	Music Information Retrieval
CBIR	Content Based Image Retrieval
SOX	Sound eXchange
ATWV	Actual Term-Weighted Value